Software Engineering Lab

202201277

Sharvil Oza

Program Inspection, Debugging and Static Analysis:

Program Inspection:

1. How many errors are there in the program? Mention the errors you have identified.

After a more detailed inspection, I've identified several issues and potential errors:

a) Category A (Data Reference Errors):

- In the constructor, there's no check if the 'prices' key exists in the data dictionary before assigning it to self.prices.

b) Category B (Data-Declaration Errors):

- The 'window' parameter is accessed from self.data['window'] in the plot_metrics method, but it's not clear where this is set.

c) Category C (Computation Errors):

- In calculate_treynor_ratio, the method uses np.log1p(abs(treynor_ratio)), which might not be mathematically correct for negative values.

d) Category D (Comparison Errors):

- In calculate_beta, the check for market_variance < 1e-8 might not be sufficient for all cases of near-zero variance.
- In calculate_rolling_treynor_ratio, there's no check for division by zero when betas are zero.

e) Category E (Control-Flow Errors):

- In plot_momentum, there's no return statement after printing "Not enough data to calculate momentum", which might lead to unexpected behavior.
- In several methods (e.g., calculate_rate_of_change, calculate_efficiency_ratio), there's no check for an empty self.prices series before accessing its elements.

f) Category F (Interface Errors):

- The add_custom_metric method catches all exceptions, which might hide important errors.

g) Category G (Input/Output Errors):

- The code assumes that certain keys exist in the data dictionary passed to the constructor (e.g., 'algorithm_period_return', 'benchmark_returns', etc.). There's no error handling if these keys are missing.
- In plot_all_metrics, there's no error handling if any of the plotting methods fail.

h) Category H (Other Checks):

- There's no input validation for the risk_free_rate parameter in the constructor.
- Some methods (e.g., calculate_sterling_ratio) use magic numbers (like 252 for annualization) without explanation.
- The class uses both pandas and numpy for similar operations, which might lead to inconsistencies.
- There's no docstring for the class itself, which would be helpful for understanding its overall purpose and usage.
- In calculate_rolling_beta, there's no handling of potential NaN values that could result from division.

Total identified errors/issues: 15

2. Which category of program inspection would you find more effective?

For this code fragment, I found Category E (Control-Flow Errors) and Category G (Input/Output Errors) to be the most effective. These categories helped identify numerous potential runtime errors that could occur due to unexpected input data, edge cases, or missing error handling. Additionally, Category H (Other Checks) proved valuable in identifying issues related to code maintainability and robustness.

3. Which type of error you are not able to identified using the program inspection?

Using program inspection alone, it's challenging to identify:

- Performance issues or inefficiencies, especially in the calculation of rolling metrics
- Logical errors in complex mathematical calculations (e.g., the correctness of financial formulas)
- Compatibility issues with different versions of the imported libraries (matplotlib, pandas, numpy, quantstats)
- Memory usage issues that might occur with very large datasets
- Floating-point precision errors that might accumulate in financial calculations

4. Is the program inspection technique worth applicable?

Yes, the program inspection technique is definitely worth applying. It has helped identify numerous potential issues that could lead to runtime errors, unexpected behaviour, or maintenance difficulties. This technique is particularly valuable for:

- Identifying potential edge cases and error conditions that might be missed in typical usage
- Ensuring consistent error handling and input validation across the class
- Detecting possible divisions by zero or other mathematical errors in financial calculations

Debugging(Part 2)

Debugging: (Submit the answers of following questions for each code fragment)

1. How many errors are there in the program? Mention the errors you have identified.

From the Debugging window, no immediate errors seem visible at this breakpoint inside the __init__ function.

However, there are a few things to check:

- Ensure that all the input data is correctly structured.
- Check if the initialized variables (market_returns, returns, prices, etc.) are correctly formatted as Pandas Series.
- Confirm that self.strategy_name, self.risk_free_rate, and self.start_date have the expected values.

2.How many breakpoints you need to fix those errors?

- To troubleshoot the __init__ function, one breakpoint at the end of the __init__ method might be sufficient to check if data initialization is correct.

a. What are the steps you have taken to fix the error you identified in the code fragment?

- Step 1: Placed a breakpoint in the __init__ function.
- Step 2: Verified that the data passed into the class was correctly processed and initialized as instance variables. This included checking:
    - market_returns
    - portfolio_value
    - risk_free_rate
    - Other time series like prices
- Step 3: Ensured the data types were consistent
- Step 4: If any errors were found with mismatched data lengths or types, corrected the data formatting either within the __init__ method or by preprocessing the data passed into the RiskMetrics class.

3.Submit your complete executable code?

```
import pandas as pd


class RiskMetrics:

    def __init__(self, data):

        self.data = data

        self.market_returns = pd.Series(data['benchmark_returns'],
index=pd.date_range(data['start_date'], periods=len(data['benchmark_returns']), freq='B'))

        self.returns = pd.Series(data['algorithm_period_return'],
index=pd.date_range(data['start_date'], periods=len(data['algorithm_period_return']),
freq='B'))

        self.portfolio_value = pd.Series(data['portfolio_value'],
index=pd.date_range(data['start_date'], periods=len(data['portfolio_value']), freq='B'))

        self.prices = pd.Series(data['prices'], index=pd.date_range(data['start_date'],
periods=len(data['prices']), freq='B'))

        self.start_date = pd.to_datetime(data['start_date'])

        self.strategy_name = data.get('strategy_name', 'RiskClass')

        self.risk_free_rate = data.get('risk_free_rate', 0.03)

        self.custom_metrics = {}

        self.output_dir = data.get('output_dir', '.')


        print("Initialized RiskMetrics class with the following data:")

        print(f"Market Returns: {self.market_returns}")

        print(f"Returns: {self.returns}")

        print(f"Portfolio Value: {self.portfolio_value}")

        print(f"Prices: {self.prices}")

        print(f"Start Date: {self.start_date}")

        print(f"Strategy Name: {self.strategy_name}")

        print(f"Risk-Free Rate: {self.risk_free_rate}")
```

Static Analysis Tools

Choose a static analysis tool (in Java, Python, C, C++) in any programming language of your interest and

identify the defects. You can also choose your own code fragment from GitHub (more than 2000 LOC)

in any programming language to perform static analysis.

Submit your results in the .xls or .jpg format only.

For this question I combined 4 python files which is over 2000 lines of code and ran pylint which is a static code analyzing tool.

Name	Date modified	Type	Size
hybridMean	09-07-2024 01:17	Python Source File	8 KB
momentumStratMulti	19-07-2024 22:16	Python Source File	17 KB
multi_support_resistance	19-07-2024 21:58	Python Source File	8 KB
pylint_output	20-10-2024 21:08	Text Document	25 KB
RiskClass	20-10-2024 20:27	Python Source File	20 KB

Now I run the following command in the command prompt and redirect the output to a text file.

```
C:\Users\hp\Desktop\RiskMetricsDebug\combined_code>pylint *.py > pylint_output.txt
```

Output:

```
************* Module hybridMean
hybridMean.py:18:27: C0303: Trailing whitespace (trailing-whitespace)
hybridMean.py:52:0: C0301: Line too long (107/100) (line-too-long)
hybridMean.py:68:0: C0303: Trailing whitespace (trailing-whitespace)
hybridMean.py:79:0: C0303: Trailing whitespace (trailing-whitespace)
hybridMean.py:143:0: C0301: Line too long (101/100) (line-too-long)
hybridMean.py:1:0: C0114: Missing module docstring (missing-module-docstring)
hybridMean.py:1:0: C0103: Module name "hybridMean" doesn't conform to snake_case naming style (invalid-name)
hybridMean.py:1:0: E0401: Unable to import 'hybrid_wrapper' (import-error)
hybridMean.py:3:0: E0401: Unable to import 'RiskClass_ticker' (import-error)
hybridMean.py:4:0: E0401: Unable to import 'stress_test' (import-error)
hybridMean.py:9:0: C0115: Missing class docstring (missing-class-docstring)
hybridMean.py:15:4: W0231: __init__ method from base class 'Strategy' is not called (super-init-not-called)
hybridMean.py:17:18: E1101: Module 'backtrader.indicators' has no 'SimpleMovingAverage' member (no-member)
hybridMean.py:19:19: E1101: Instance of 'tuple' has no 'period' member (no-member)
hybridMean.py:25:4: C0116: Missing function or method docstring (missing-function-docstring)
hybridMean.py:35:45: W0212: Access to a protected member _name of a client class (protected-access)
hybridMean.py:38:46: W0212: Access to a protected member _name of a client class (protected-access)
hybridMean.py:44:59: W0212: Access to a protected member _name of a client class (protected-access)
hybridMean.py:52:41: W0212: Access to a protected member _name of a client class (protected-access)
hybridMean.py:62:45: E1101: Instance of 'tuple' has no 'offset' member (no-member)
hybridMean.py:73:47: E1101: Instance of 'tuple' has no 'offset' member (no-member)
hybridMean.py:41:12: W0201: Attribute 'bar_executed' defined outside __init__ (attribute-defined-outside-init)
hybridMean.py:46:8: W0201: Attribute 'order' defined outside __init__ (attribute-defined-outside-init)
hybridMean.py:99:0: C0103: Constant name "multiplier" doesn't conform to UPPER_CASE naming style (invalid-name)
hybridMean.py:100:0: C0103: Constant name "timespan" doesn't conform to UPPER_CASE naming style (invalid-name)
hybridMean.py:101:0: C0103: Constant name "backtest_start_date" doesn't conform to UPPER_CASE naming style (invalid-name)
hybridMean.py:102:0: C0103: Constant name "backtest_end_date" doesn't conform to UPPER_CASE naming style (invalid-name)
hybridMean.py:103:0: C0103: Constant name "api_key" doesn't conform to UPPER_CASE naming style (invalid-name)
hybridMean.py:104:0: C0103: Constant name "local_dir" doesn't conform to UPPER_CASE naming style (invalid-name)
hybridMean.py:105:0: C0103: Constant name "trade_ticker" doesn't conform to UPPER_CASE naming style (invalid-name)
hybridMean.py:133:6: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
hybridMean.py:136:6: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
hybridMean.py:3:0: C0411: third party import "RiskClass_ticker.RiskMetricsTicker" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
hybridMean.py:4:0: C0411: third party import "stress_test.StressTesting" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
hybridMean.py:5:0: C0411: third party import "backtrader" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
hybridMean.py:6:0: C0411: third party import "numpy" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
hybridMean.py:7:0: C0411: third party import "pandas" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
```

```
************* Module momentumStratMulti
momentumStratMulti.py:50:0: C0303: Trailing whitespace (trailing-whitespace)
momentumStratMulti.py:69:0: C0301: Line too long (122/100) (line-too-long)
momentumStratMulti.py:72:0: C0301: Line too long (123/100) (line-too-long)
momentumStratMulti.py:85:0: C0301: Line too long (108/100) (line-too-long)
momentumStratMulti.py:88:0: C0303: Trailing whitespace (trailing-whitespace)
momentumStratMulti.py:94:0: C0303: Trailing whitespace (trailing-whitespace)
momentumStratMulti.py:105:0: C0303: Trailing whitespace (trailing-whitespace)
momentumStratMulti.py:107:0: C0301: Line too long (125/100) (line-too-long)
momentumStratMulti.py:108:0: C0301: Line too long (123/100) (line-too-long)
momentumStratMulti.py:113:0: C0301: Line too long (107/100) (line-too-long)
momentumStratMulti.py:116:0: C0301: Line too long (109/100) (line-too-long)
momentumStratMulti.py:117:0: C0301: Line too long (104/100) (line-too-long)
momentumStratMulti.py:120:0: C0301: Line too long (107/100) (line-too-long)
momentumStratMulti.py:121:0: C0301: Line too long (119/100) (line-too-long)
momentumStratMulti.py:124:0: C0301: Line too long (107/100) (line-too-long)
momentumStratMulti.py:125:0: C0301: Line too long (137/100) (line-too-long)
momentumStratMulti.py:193:0: C0301: Line too long (108/100) (line-too-long)
momentumStratMulti.py:1:0: C0114: Missing module docstring (missing-module-docstring)
momentumStratMulti.py:1:0: C0103: Module name "momentumStratMulti" doesn't conform to snake_case naming style (invalid-name)
momentumStratMulti.py:13:0: E0401: Unable to import 'hybrid_wrapper' (import-error)
momentumStratMulti.py:15:0: E0401: Unable to import 'stress_test' (import-error)
momentumStratMulti.py:16:0: E0401: Unable to import 'RiskClass_ticker' (import-error)
momentumStratMulti.py:17:0: W0404: Reimport 'letter' (imported line 8) (reimported)
momentumStratMulti.py:23:0: C0115: Missing class docstring (missing-class-docstring)
momentumStratMulti.py:23:0: R0902: Too many instance attributes (9/7) (too-many-instance-attributes)
momentumStratMulti.py:34:4: W0231: __init__ method from base class 'Strategy' is not called (super-init-not-called)
momentumStratMulti.py:45:76: E1101: Instance of 'tuple' has no 'rsi_period' member (no-member)
momentumStratMulti.py:46:36: E1121: Too many positional arguments for constructor call (too-many-function-args)
momentumStratMulti.py:46:36: E1123: Unexpected keyword argument 'period_me1' in constructor call (unexpected-keyword-arg)
momentumStratMulti.py:46:36: E1123: Unexpected keyword argument 'period_me2' in constructor call (unexpected-keyword-arg)
momentumStratMulti.py:46:36: E1123: Unexpected keyword argument 'period_signal' in constructor call (unexpected-keyword-arg)
momentumStratMulti.py:47:50: E1101: Instance of 'tuple' has no 'macd1' member (no-member)
momentumStratMulti.py:48:50: E1101: Instance of 'tuple' has no 'macd2' member (no-member)
momentumStratMulti.py:49:53: E1101: Instance of 'tuple' has no 'macdsig' member (no-member)
momentumStratMulti.py:58:4: C0116: Missing function or method docstring (missing-function-docstring)
momentumStratMulti.py:68:33: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:69:45: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:71:33: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:72:46: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:77:59: W0212: Access to a protected member _name of a client class (protected-access)
```

```
momentumStratMulti.py:72:46: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:77:59: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:79:19: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:85:41: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:92:26: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:96:44: E1101: Instance of 'tuple' has no 'cash_factor' member (no-member)
momentumStratMulti.py:103:46: E1101: Instance of 'tuple' has no 'volume_factor' member (no-member)
momentumStratMulti.py:107:32: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:107:82: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:107:101: E1101: Instance of 'tuple' has no 'stop_loss' member (no-member)
momentumStratMulti.py:108:52: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:109:27: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:110:33: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:113:28: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:113:59: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:113:88: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:114:47: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:116:30: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:116:61: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:116:90: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:117:56: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:120:28: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:120:59: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:120:88: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:121:48: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:124:28: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:124:59: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:124:88: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:125:61: W0212: Access to a protected member _name of a client class (protected-access)
momentumStratMulti.py:136:15: W0621: Redefining name 'ticker' from outer scope (line 215) (redefined-outer-name)
momentumStratMulti.py:74:12: W0201: Attribute 'bar_executed' defined outside __init__ (attribute-defined-outside-init)
momentumStratMulti.py:142:0: C0103: Constant name "multiplier" doesn't conform to UPPER_CASE naming style (invalid-name)
momentumStratMulti.py:143:0: C0103: Constant name "timespan" doesn't conform to UPPER_CASE naming style (invalid-name)
momentumStratMulti.py:144:0: C0103: Constant name "backtest_start_date" doesn't conform to UPPER_CASE naming style (invalid-name)
momentumStratMulti.py:145:0: C0103: Constant name "backtest_end_date" doesn't conform to UPPER_CASE naming style (invalid-name)
momentumStratMulti.py:146:0: C0103: Constant name "api_key" doesn't conform to UPPER_CASE naming style (invalid-name)
momentumStratMulti.py:147:0: C0103: Constant name "local_dir" doesn't conform to UPPER_CASE naming style (invalid-name)
momentumStratMulti.py:148:0: C0103: Constant name "trade_ticker" doesn't conform to UPPER_CASE naming style (invalid-name)
momentumStratMulti.py:149:0: C0103: Constant name "budget" doesn't conform to UPPER_CASE naming style (invalid-name)
momentumStratMulti.py:177:6: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
momentumStratMulti.py:180:6: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
```

```
momentumStratMulti.py:180:6: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
momentumStratMulti.py:329:0: C0116: Missing function or method docstring (missing-function-docstring)
momentumStratMulti.py:329:0: R0914: Too many local variables (22/15) (too-many-locals)
momentumStratMulti.py:329:22: W0621: Redefining name 'folder_name' from outer scope (line 262) (redefined-outer-name)
momentumStratMulti.py:329:0: R0915: Too many statements (56/50) (too-many-statements)
momentumStratMulti.py:10:0: C0411: standard import "os" should be placed before third party imports "matplotlib.pyplot", "reportlab.platypus.SimpleDocTemplate",
"reportlab.lib.styles.getSampleStyleSheet", "reportlab.lib.colors", "reportlab.lib.pagesizes.letter", "pandas" (wrong-import-order)
momentumStratMulti.py:15:0: C0411: third party import "stress_test.StressTesting" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
momentumStratMulti.py:16:0: C0411: third party import "RiskClass_ticker.RiskMetricsTicker" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
momentumStratMulti.py:17:0: C0411: third party import "reportlab.lib.pagesizes.letter" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
momentumStratMulti.py:18:0: C0411: third party import "reportlab.pdfgen.canvas" should be placed before first party import "RiskClass.RiskMetrics"  (wrong-import-order)
momentumStratMulti.py:19:0: C0411: standard import "sys" should be placed before third party imports "matplotlib.pyplot", "reportlab.platypus.SimpleDocTemplate",
"reportlab.lib.styles.getSampleStyleSheet" (...) "RiskClass_ticker.RiskMetricsTicker", "reportlab.lib.pagesizes.letter", "reportlab.pdfgen.canvas" and first party import
"RiskClass.RiskMetrics"  (wrong-import-order)
momentumStratMulti.py:17:0: C0412: Imports from package reportlab are not grouped (ungrouped-imports)
momentumStratMulti.py:18:0: W0611: Unused canvas imported from reportlab.pdfgen (unused-import)
************* Module multi_support_resistance
multi_support_resistance.py:22:0: C0301: Line too long (111/100) (line-too-long)
multi_support_resistance.py:23:0: C0301: Line too long (105/100) (line-too-long)
multi_support_resistance.py:37:0: C0301: Line too long (117/100) (line-too-long)
multi_support_resistance.py:41:0: C0301: Line too long (118/100) (line-too-long)
multi_support_resistance.py:52:0: C0301: Line too long (108/100) (line-too-long)
multi_support_resistance.py:185:0: C0301: Line too long (101/100) (line-too-long)
multi_support_resistance.py:1:0: C0114: Missing module docstring (missing-module-docstring)
multi_support_resistance.py:4:0: E0401: Unable to import 'hybrid_wrapper' (import-error)
multi_support_resistance.py:7:0: C0115: Missing class docstring (missing-class-docstring)
multi_support_resistance.py:7:0: C0103: Class name "multi_support_resistance" doesn't conform to PascalCase naming style (invalid-name)
multi_support_resistance.py:16:4: W0231: __init__ method from base class 'Strategy' is not called (super-init-not-called)
multi_support_resistance.py:25:4: C0116: Missing function or method docstring (missing-function-docstring)
multi_support_resistance.py:35:34: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:37:40: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:39:34: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:41:41: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:44:59: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:46:20: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:52:41: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:59:31: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:66:38: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:74:4: C0116: Missing function or method docstring (missing-function-docstring)
multi_support_resistance.py:87:38: W0212: Access to a protected member _name of a client class (protected-access)
```

```
multi_support_resistance.py:74:4: C0116: Missing function or method docstring (missing-function-docstring)
multi_support_resistance.py:82:38: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:83:36: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:106:4: C0116: Missing function or method docstring (missing-function-docstring)
multi_support_resistance.py:107:29: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:108:53: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:111:4: C0116: Missing function or method docstring (missing-function-docstring)
multi_support_resistance.py:116:4: C0116: Missing function or method docstring (missing-function-docstring)
multi_support_resistance.py:122:41: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:123:20: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:125:4: C0116: Missing function or method docstring (missing-function-docstring)
multi_support_resistance.py:131:43: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:132:20: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:138:31: W0212: Access to a protected member _name of a client class (protected-access)
multi_support_resistance.py:141:0: C0103: Constant name "multiplier" doesn't conform to UPPER_CASE naming style (invalid-name)
multi_support_resistance.py:142:0: C0103: Constant name "timespan" doesn't conform to UPPER_CASE naming style (invalid-name)
multi_support_resistance.py:143:0: C0103: Constant name "backtest_start_date" doesn't conform to UPPER_CASE naming style (invalid-name)
multi_support_resistance.py:144:0: C0103: Constant name "backtest_end_date" doesn't conform to UPPER_CASE naming style (invalid-name)
multi_support_resistance.py:145:0: C0103: Constant name "api_key" doesn't conform to UPPER_CASE naming style (invalid-name)
multi_support_resistance.py:146:0: C0103: Constant name "local_dir" doesn't conform to UPPER_CASE naming style (invalid-name)
multi_support_resistance.py:147:0: C0103: Constant name "trade_ticker" doesn't conform to UPPER_CASE naming style (invalid-name)
multi_support_resistance.py:169:6: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
multi_support_resistance.py:172:6: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
multi_support_resistance.py:3:0: W0611: Unused pandas imported as pd (unused-import)
multi_support_resistance.py:5:0: W0611: Unused RiskMetrics imported from RiskClass (unused-import)
************* Module RiskClass
RiskClass.py:15:0: C0301: Line too long (139/100) (line-too-long)
RiskClass.py:57:0: C0301: Line too long (131/100) (line-too-long)
RiskClass.py:66:0: C0301: Line too long (106/100) (line-too-long)
RiskClass.py:118:0: C0301: Line too long (106/100) (line-too-long)
RiskClass.py:241:0: C0301: Line too long (103/100) (line-too-long)
RiskClass.py:349:0: C0301: Line too long (103/100) (line-too-long)
RiskClass.py:1:0: C0114: Missing module docstring (missing-module-docstring)
RiskClass.py:1:0: C0103: Module name "RiskClass" doesn't conform to snake_case naming style (invalid-name)
RiskClass.py:9:0: C0115: Missing class docstring (missing-class-docstring)
RiskClass.py:9:0: R0902: Too many instance attributes (9/7) (too-many-instance-attributes)
RiskClass.py:10:23: W0621: Redefining name 'data' from outer scope (line 555) (redefined-outer-name)
RiskClass.py:123:8: W0612: Unused variable 'fig' (unused-variable)
RiskClass.py:219:15: W0718: Catching too general exception Exception (broad-exception-caught)
RiskClass.py:342:15: E1123: Unexpected keyword argument 'cutoff' in function call (unexpected-keyword-arg)
```

```
RiskClass.py:363:15: E1101: Module 'quantstats.stats' has no 'conditional_sharpe' member (no-member)
RiskClass.py:462:15: E1123: Unexpected keyword argument 'cutoff' in function call (unexpected-keyword-arg)
RiskClass.py:489:15: E1101: Module 'quantstats.stats' has no 'upside_capture' member (no-member)
RiskClass.py:498:15: E1101: Module 'quantstats.stats' has no 'downside_deviation' member (no-member)
RiskClass.py:516:15: E1101: Module 'quantstats.stats' has no 'jensens_alpha' member (no-member)
RiskClass.py:525:15: E1101: Module 'quantstats.stats' has no 'upside_capture' member (no-member)
RiskClass.py:534:15: E1101: Module 'quantstats.stats' has no 'downside_capture' member (no-member)
RiskClass.py:9:0: R0904: Too many public methods (35/20) (too-many-public-methods)
RiskClass.py:546:0: W0105: String statement has no effect (pointless-string-statement)
RiskClass.py:1:0: R0801: Similar lines in 2 files
==hybridMean:[143:165]
==momentumStratMulti:[193:215]
length_difference = abs(len(benchmark_returns) - len(returns))

print("\n----- Time Period -----")
# Calculate risk metrics
risk_metrics = RiskMetrics(
    {
        'algorithm_period_return': returns,
        'benchmark_returns': benchmark_returns,
        'portfolio_value': portfolio_values,
        'threshold_returns': 0.03,
        'prices': combined_df[combined_df['ticker'] == trade_ticker]['close'].values,
        'window': 20,
        'length_difference': length_difference,
        'start_date': backtest_start_date
    }
)

print("\n----- Portfolio Analytics -----")
risk_metrics.plot_all_metrics()
# Calculate risk metrics for individual tickers
print("\n----- Risk Metrics for Individual Tickers -----")
for ticker in tickers: (duplicate-code)
RiskClass.py:1:0: R0801: Similar lines in 2 files
==hybridMean:[180:200]
==momentumStratMulti:[229:249]
            'window': 20,
            'length_difference': length_difference,
            'start_date': backtest_start_date
        },
```

```
# Plot the result
data_wrapper.cerebro.plot()

# Perform stress testing
print("\n--- Stress Testing Results ---")

# Calculate returns
portfolio_values = pd.Series(strat.portfolio_values)
returns = portfolio_values.pct_change().dropna().values

# Convert dates to datetime if they're not already (duplicate-code)
RiskClass.py:1:0: R0801: Similar lines in 2 files
==hybridMean:[107:125]
==momentumStratMulti:[151:169]
data_wrapper = HybridDataWrapper(
    tickers=tickers,
    multiplier=multiplier,
    timespan=timespan,
    backtest_start_date=backtest_start_date,
    backtest_end_date=backtest_end_date,
    api_key=api_key,
    local_dir=local_dir,
    trade_ticker=trade_ticker
)

# Fetch and transform data
combined_df = data_wrapper.fetch_and_transform_data()

# Add the data to cerebro
data_wrapper.add_data_to_cerebro(combined_df)

# Add the strategy to Cerebro (duplicate-code)
RiskClass.py:1:0: R0801: Similar lines in 2 files
==hybridMean:[167:179]
==momentumStratMulti:[216:228]
    ticker_returns = np.diff(ticker_prices) / ticker_prices[:-1]

    benchmark_returns = combined_df[combined_df['ticker']
                                    == 'SPY']['close'].pct_change().dropna().values
    length_difference = abs(len(benchmark_returns) - len(ticker_returns))
```

```
    risk_metrics_ticker = RiskMetricsTicker(
        {
            'algorithm_period_return': ticker_returns,
            'benchmark_returns': benchmark_returns,
            'portfolio_value': ticker_prices,
            'threshold_returns': 0.03, (duplicate-code)
RiskClass.py:1:0: R0801: Similar lines in 2 files
==momentumStratMulti:[182:192]
==multi_support_resistance:[174:184]
strat = strats[0]
portfolio_values = strat.portfolio_values
portfolio_dates = strat.portfolio_dates

# Ensure portfolio_values is not empty before calculating returns
if len(portfolio_values) > 1:
    returns = np.diff(portfolio_values) / portfolio_values[:-1]
else:
    returns = []
 (duplicate-code)
RiskClass.py:1:0: R0801: Similar lines in 2 files
==hybridMean:[130:140]
==momentumStratMulti:[174:184]
print("\n----- Strategy Execution  -----")
# Print out the starting conditions
print('Starting Portfolio Value: %.2f' % data_wrapper.cerebro.broker.getvalue())

strats = data_wrapper.cerebro.run()
print('Final Portfolio Value: %.2f' % data_wrapper.cerebro.broker.getvalue())

# Access the portfolio values logged during the strategy execution
strat = strats[0]
portfolio_values = strat.portfolio_values (duplicate-code)
RiskClass.py:1:0: R0801: Similar lines in 2 files
==hybridMean:[25:34]
==momentumStratMulti:[58:67]
        dt = dt or self.datas[0].datetime.date(0)
        print(f'{dt.isoformat()} {txt}')
```

```
    def notify_order(self, order):
        if order.status in [order.Submitted, order.Accepted]:
            return

        if order.status in [order.Completed]:
            if order.isbuy(): (duplicate-code)

 ------------------------------------
 Your code has been rated at 5.38/10
```